VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019 Lukáš Ciahotný

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Ciahotný**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: profiq s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Dohnálek**

Konzultant bakalářské práce: Ing. Ondřej Fuchsík

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

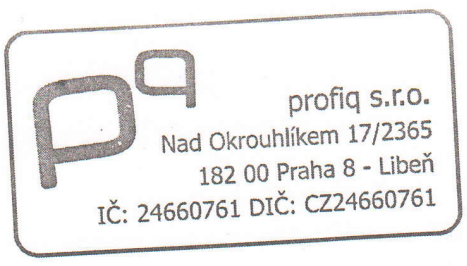
prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 11. dubna 2019 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 11. dubna 2019 .....



Rád bych na tomto místě poděkoval svému vedoucímu Ing. Dohnálkovi za cenné rady při sepisování této práce, firmě profiq s.r.o., která mi poskytla tuto příležitost pro sepsání této práce a jmenovitě Ing. Fuchsíkovi za jeho podporu a vedení v průběhu praxe.

Abstrakt

Tato bakalářská práce popisuje studentovu odbornou praxi ve společnosti profiq s.r.o. Na začátku je popsána společnost a pracovní zařazení studenta, dále v textu je popsáno řešení jednotlivých zadaných úkolů během bakalářské praxe. Student měl během práce na starost vývoj automatizovaného obnovování SSL certifikátů pro subdomény ve firemním intranetu. Dále měl na starost vytvoření automatického testování backend API. V bakalářské práci jsou popsány problémy, které nastaly při vypracování obou úkolů a popis jejich řešení. V závěru práce popisuje použité znalosti během studia a technologie a postupy, ve kterých se student zlepšil. Na konci práce je celkové zhodnocení praxe.

Klíčová slova: Odborná praxe, JavaScript, Python, Bash, Selenium Webdriver, Gitlab CI/CD, CA, SSH, Kontrola kvality, Nginx, Docker

Abstract

This bachelor thesis describes student's professional practice in profiq s.r.o. At the beginning is described the company and working position of the student, further in the text is described solution of individual student's tasks during bachelor practice. During students work, the student is in charge of developing an automated recovery of SSL certificates for subdomains in the corporate intranet. He was also in charge of creating automatic backend API testing. The bachelor thesis describes the problems that occurred in the elaboration of both tasks and a description of their solution. At the end of the work student describe used knowledge during the study and technology and procedures in which the student improved. At the end of the work there is an overall evaluation of the practice.

Key Words: Professional Practice, JavaScript, Python, Bash, Selenium Webdriver, Gitlab CI/CD, CA, SSH, Quality assurance, Nginx, Docker

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CI	–	Continuous integration
CD	–	Continuous delivery
CA	–	Certificate authority
JS	–	JavaScript
DNS	–	Domain Name System
TTL	–	Time to live
CSS	–	Cascading Style Sheets
LDAP	–	Lightweight Directory Access Protocol

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce popisuje studentovu odbornou praxi ve společnosti profiq s.r.o. Společnost se v době studentova nástupu věnovala hlavně poskytováním služeb v oblasti testování.

Student si prvně musel nastudovat technologii Docker a nahlédnout do infrastruktury firmy a poté mu byl přiřazený úkol na zautomatizování procesu pro vytváření a obnovování SSL certifikátů s využitím služby Let's encrypt. Firma dosud měla zabezpečenou doménu pomocí automatického řešení od poskytovatele, ale chtěla si zabezpečit komunikaci certifikáty i na subdoménách v jejich intranetu. Celý proces měl být zautomatizován a nasazen na GitLab CI/CD, aby bylo možné spouštět proces pravidelně z důvodu 3 měsíční životnosti SSL certifikátu. Při vytváření skriptů a nastavování GitLab CI/CD student narazil na několik problémů týkajících se využití nástroje Webdrive.IO a změny GitLab API pro stahování artefaktů během vypracování praxe.

Studentův druhý úkol bylo seznámení s firemní LDAP backend API. Následně si musel přečíst dokumentaci ke knihovnám unittest a requests pro jazyk Python. Po seznámení a vyzkoušení bylo studentovi zadáno vytvořit automatické testy pro backend API, které se budou spouštět při každém Merge requestu do hlavního repozitáře projektu. Testování spočívalo v kontrole stavových kódů při odpovědi serveru na klientské požadavky. Při vypracovávání úkolu narazil student na problémy, které musel řešit vytvářením pomocných funkcí, aby bylo možné správně otestovat výsledky.

2 Popis odborného zaměření firmy a pracovní zařazení studenta

Součastí této kapitoly je popis společnosti profiq s.r.o., ve které student vypracovával svou odbornou praxi a dále studentovo zaměření ve zmíněné firmě.

2.1 Společnost profiq s.r.o.

Společnost profiq s.r.o. se zaměřuje na poskytování služeb softwarového inženýrství v oblasti vývoje a testování. Své služby zaměřuje hlavně na americký trh v Silicon Valley pomocí agilního způsobu vývoje, který funguje na rychlém vývoji a testováním, následným předložením zákazníkovi a úpravami podle zákazníkovy zpětné vazby. [1]

V součastnosti společnost zaměstnává zhruba 50 zaměstnanců a stále roste. Firma své služby poskytuje v různých formách - od poskytování celých týmů na komplexní pokrytí určité oblasti vývoje, tak i jednotlivé specialisty na konkrétní úkoly. Mezi hlavní zákazníky patří firmy jako ForgeRock, která poskytuje software pro správu identit. LifeRay, která se zabývá vývojem portálových řešení a firmou SlamData, zabývající se jednodušší integrací dat ze služeb S3, Azure a Google cloudu.

2.2 Pracovní zaměření studenta ve firmě profiq s.r.o.

Ve firmě student pracoval na pozici softwarového inženýra. V rámci pracovního zařazení na bakalářské praxi mu byly přiřazeny dva úkoly.

První úkol bylo zautomatizování vytváření a nasazení SSL certifikátu s využitím služby Let's Encrypt a následné nasazení na Gitlab CI/CD. Problém v tomto úkolu byl, že firma nemá 100% kontrolu nad serverem, kde běží doména, a tak nemohl být použít přístup přes Shell. Studentovým úkolem bylo prozkoumat možnosti, jak využít službu Let's Encrypt a analyzovat, jakými programovacími jazyky a nástroji nejlépe dosáhnout automatického vytváření a nasazování SSL certifikátů. Záměrem bylo vytvořit řešení, které se bude používat dlouhodobě.

Druhý úkol bylo automatické testování backend API. Na tomto úkolu bylo potřeba zorientovat se v projektu a následně vymyslet testovací scénáře, které by pokryly kontrolu základních funkcí. Po vymyšlení testovacích scénářů mohl student začít proces automatizace testů. Během procesu automatizace narazil student na problémy, které musel řešit vytvářením dodatečných funkcí. Záměrem bylo vytvořit řešení, které by bylo fungovalo dlouhodobě, a nad kterým by bylo možné stavět a vytvářet dodatečné testy.

3 Seznam zadaných úloh studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti

V následujících dvou kapitolách jsou lehce nastíněné kroky, které popisují práci na dvou úkolech, jenž byly studentovi přiděleny jako náplň práce v průběhu odborné praxe ve společnosti profiq s.r.o.

3.1 Automatizace obnovení SSL certifikátů s využitím služby Let's Encrypt

V podsekcích této kapitoly jsou sepsány části prvního studentova úkolu na vytvoření automatického obnovování SSL certifikátů s využitím služeb Let's Encrypt.

3.1.1 Seznámení s CertBot

Studentovým prvním úkolem bylo se seznámit se službou Let's Encrypt a jejím klientem CertBot. Zjistit co služba nabízí, jakým způsobem funguje a jak ji zapojit do svého řešení. Získávání celkového přehledu o službě. Následně bylo potřeba se prakticky seznámit se službou, prozkoumat uživatelskou dokumentaci a prakticky si vyzkoušet nabyté dovednosti. Na celý tenhle proces, ve kterém se student seznamoval a zkoušel pracovat se službou Let's Encrypt a jejím klientem CertBot, bylo vyhrazeno 5 dní.

3.1.2 Vytváření CertBot skriptu

Pro začátek bylo potřeba vytvořit skript, který by vytvořil hodnoty pro DNS výzvy, které se budou v dalším kroku nahrávat na doménový server kvůli ověření vlastnictví domény. Problém nastal v momentě, kdy student potřeboval vygenerovat certifikát pro více domén najednou. Proces vytváření a testování CertBot skriptu zabralo studentovi 7 dní z důvodu hodinového čekání na TTL na doménovém serveru po každé úpravě kódu.

3.1.3 Začátek automatizace

Pro tento úkol bylo potřeba vybrat si správný nástroj pro automatické nasazování DNS výzvy na webovou stránku DNS serveru. Jako první si student vybral JavaScript s rozšířením WebDriver.io, které se ukázalo jako špatný krok a bylo potřeba předělávat vytvořený kód na Selenium-WebDriver, se kterým už bylo dosaženo požadovaných cílů. Kvůli špatně zvolené technologii a následnému předělávání kódu bylo na automatizaci využito 10 dní.

3.1.4 Nasazení do Gitlab CI/CD

Pro poslední úkol v dané úloze bylo potřeba si přečíst dokumentaci Gitlab CI/CD a prakticky se seznámit se službou. Dále bylo potřeba převést již vytvořené skripty do Gitlabu, vytvoření gitlab-ci.yml souboru. Posléze bylo potřeba vytvořit artefakt s certifikátem. Následovalo předání

artefaktu pro nasazení certifikátu na server a testování funkčnosti. Nakonec bylo potřeba vytvořit Shell skripty na dvou serverech, které si přes SSH stáhnou artefakt s certifikátem a přepíšou stávající certifikát a restartují server. Na nejnáročnější úkol, který byl přidělen studentovi, bylo vyhrazeno 15 dní, a to kvůli studentově absenci znalostí s Gitlab CI/CD a technologii Docker, které se byl nucen doučit. Jako další zdržení byl prosincový update Gitlabu, který změnil přistupování na poslední vytvořený artefakt přes rozhraní API a kvůli němu bylo nutné upravit stávající řešení.

3.2 Automatické testování backend API

V podsekcích této kapitoly jsou sepsány části druhého studentova úkolu na vytvoření automatizovaného testování backend API.

3.2.1 Seznámení s projektem

Studentovým prvním úkolem bylo seznámit se s projektem pro evidenci zaměstnanců LDAP. Zjistit co projekt nabízí, jakým způsobem funguje a jaké testy by bylo možné realizovat. Pro toto seznámení bylo vyhrazeno 5 pracovních dní.

3.2.2 Provedení manuálního průzkumného testování

Před zahájením automatizace bylo nutné, aby si student zkusil vymyslet testovací případy a následně je vyzkoušel provést manuálně. To z důvodu zjištění, jak se jednotlivé komponenty chovají za určitých podmínek. Na základě výsledků z manuálních testů byl studentův úkol předělat nebo vylepšit testy, aby bylo jednodušší odhalit chyby na základní funkčnosti, které by při změnách kódu mohly nastat. Pro vyzkoušení testů manuálně a následnou úpravu testovacích scénářů bylo vyhrazeno 5 dní.

3.2.3 Vytváření automatických testů

Automatizace testů byla hlavní náplní studenta v tomto zadání. Jako první úkol pro studenta bylo vybrat vhodný nástroj pro psaní testů a poté se v něm naučit pracovat. Dalším úkolem bylo zjistit, jaké možnosti nástroj nabízí a následně začít s automatizací. Při psaní automatických testů se vyskytly problémy, které student musel řešit vytvářením dodatečných funkcí, aby bylo možné řádně otestovat výsledky. Tenhle proces vytváření automatických testů a tvoření dodatečných funkcí, které byly nutné pro otestování správnosti výsledků testů, zabral studentovi zbytek pracovních dnů na odborné praxi.

4 Zvolený postup řešení zadaných úkolů

V následujících dvou kapitolách je možné najít postupy řešení dvou úkolů, které byly zadány studentovi. Tyto kapitoly jsou více zaměřeny na výsledné řešení úkolů, které student měl za úkol vykonat a dále jsou detailněji popsány v ukázkách kódů.

4.1 Automatizace obnovení SSL certifikátů s využitím služby Let's encrypt

Tahle kapitola je zaměřena na popis řešení prvního studentova úkolu na vytvoření automatického vytváření SSL certifikátů a nasazení SSL certifikátů s využitím služby Let's Encrypt. V jednotlivých částech této kapitoly je popsáno, jak student postupoval, a na jaké problémy v tomto procesu narazil a jak je řešil.

4.1.1 Seznámení s CertBot

Ten, kdo certifikáty vydává, je běžně označován jako tzv. certifikační autorita, zkratkou CA. [2] Pro tento úkol byla vybrána certifikační autorita Let's Encrypt a její nástroj pro získávání certifikátů nazývaný CertBot. Tenhle nástroj poskytuje vynikající automatizované pluginy pro různý hosting a různé typy serverů. Aby bylo možné získat certifikát pro doménu od Let's Encrypt, musíte prokázat, že máte kontrolu nad danou doménou, pro kterou chcete vytvořit certifikát. Aby toho bylo docíleno je potřeba použít software, který používá ACME protokol, který typicky běží na webovém hostiteli.

K tomu je potřeba vědět, zda máte Shell přístup (známý též jako SSH přístup) k vašemu webovému hostiteli. Pokud spravujete své webové stránky přímo skrz ovládací panel jako je cPanel, Plesk nebo WordPress je dosti velká šance, že nemáte Shell přístup. [3]

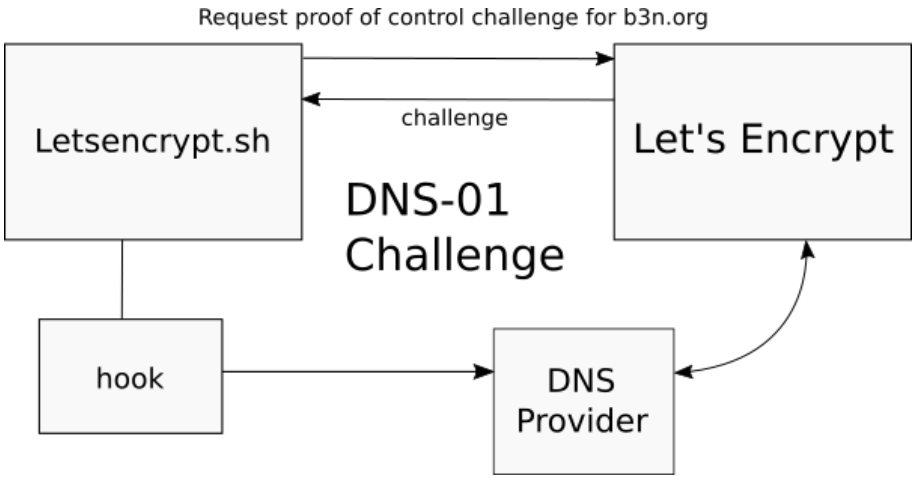
S Shell přístupem: Pokud máte Shell přístup, tak Let's Encrypt silně doporučuje použít ACME klienta CertBot, který zautomatizuje vydávání a instalaci certifikátů. Klient má taky expertní mód pro uživatele, kteří nechtějí výchozí konfiguraci. Je lehký k použití, funguje na více operačních systémech a má rozsáhlou dokumentaci

Bez Shell přístupu: Asi nejlepším způsobem pro použití Let's Encrypt bez Shell přístupu je integrovaná podpora od poskytovatele hostingu. Pokud váš poskytovatel hostingu podporuje Let's Encrypt, můžete požádat o bezplatný certifikát vaším jménem, nainstalovat ho a aktualizovat jej automaticky. Pro některé poskytovatele hostingu je toto nastavení konfigurace nutné zapnout. Ostatní poskytovatelé hostingu automaticky žádají a instalují certifikáty pro všechny své zákazníky.

V případě studentova úkolu byl problém, že firma nemá 100% Shell přístup k doménovému poskytovateli (firma má řízené hostování). Firma má Let's Encrypt certifikát nasazený na doménu jako automatické řešení od poskytovatele, ale chtěla by vygenerovat a nasadit certifikát i pro subdomény na vnitřní síti, což už neposkytuje poskytovatel hostingu. Naštěstí existuje řada pluginů, které CertBot nabízí a které řeší takové problémy. Jeden z nich se nazývá **manual**.

Tenhle plugin by měl být schopný vytvořit certifikát pro každého poskytovatele. I přesto, že se plugin nazývá **manual**, používá se pro vytváření vlastního automatického procesu pro získávání certifikátů. [4]

Jako první je potřeba si vybrat druh výzvy pro vygenerování certifikátu. Výzva je nutná k ověření kontroly nad doménou, kde se generuje certifikát. Manual plugin může použít HTTP, DNS nebo TLS-SNI výzvu. Za použití "–preferred-challenges" je možné si vybrat výzvu podle uvážení. Více o výzvách je možné k dočtení v dokumentaci. [5] Pro tuto práci se bude používat DNS výzva z důvodu blokace HTTP výzvy ze strany poskytovatele. Při použití DNS výzvy bude potřeba umístit TXT DNS záznam se specifickým kódem pod firemní doménové jméno. Více k DNS výzvě je možné vidět na obrázku.1



Obrázek 1: DNS výzva

4.1.2 Vytváření CertBot skriptu

Příkaz, který slouží k vygenerování certifikátu je možno vidět ve výpisu:1

```
...
certbot certonly --manual --no-eff-email --preferred-challenges dns --manual-
    auth-hook
$CURRENT_PATH/authenticator.sh --server https://acme-v02.api.letsencrypt.org/
    directory
--cert-name profiq.com -d *.profiq.com -d api.ldap.profiq.com --agree-tos
--manual-public-ip-logging-ok -m lukas.ciahotny@profiq.com
...
```

Výpis 1: CertBot příkaz

Dále jsou vysvětlené jednotlivé parametry, které se používají v ukázce příkazu. Parametr **Server https://acme-v02.api.letsencrypt.org/directory** se zadává pokud potřebujete vy-

tvořit certifikát s hvězdičkovým zápisem jako se používá pro toto řešení (-d *.profiq.com), je třeba specifikovat rozhraní API verze 2. Jestliže není třeba vytvořit certifikát za pomoci hvězdičkového zápisu, tak není třeba zadávat parametr server vůbec. Je dobré podotknout, že když se testuje, zda se certifikát vygeneroval správně, je vhodné použít server https://acme-staging-v02.api.letsencrypt.org/directory, který slouží pro testování a neodečítá certifikáty z týdenního limitu, který má Let's Encrypt nastavený na 20 certifikátů za týden. Další parametr v příkazu je **d *.profiq.com -d api.ldap.profiq.com**, který slouží k výběru domény, pro kterou se bude vytvářet certifikát. Pokud je potřeba vytvořit certifikát pro více domén najednou, lze tento parametr řetězit tímto způsobem: -d *.profiq.com -d api.ldap.profiq.com nebo tímto: -d *.profiq.com, api.ldap.profiq.com. Ačkoli se nedoporučuje generovat certifikát pro více domén najednou z důvodu bezpečnosti. Pokud se podaří útočníkovi odcizit takový certifikát, může napodobit jakoukoliv z domén pro kterou byl certifikát vystaven. **cert-name profiq.com** slouží jako jméno finálního certifikátu. A nakonec nejdůležitější z parametrů **manual-auth-hook $CURRENT_PATH/authenticator.sh**, který obsahuje cestu k authenticator.sh skriptu, který se spustí před validací kontroly vlastnictví domény. Skript obdrží několik proměnných, které lze ve skriptu využít. Nejdůležitější proměnná je CERTBOT_VALIDATION, která obsahuje validační řetězec, který je nutný vložit do nastavení DNS na doméně a CERTBOT_DOMAIN, která obsahuje jméno, pod kterým se má uložit validační řetězec. Více o manual-hook příkazu bude student rozebírat v podsekci Začátek automatizace.

4.1.3 Začátek automatizace

Uvnitř manuálního auth hook skriptu je potřeba nastavit výzvy na doménu. Authenticator.sh skript je možné vidět na ukázce kódu 2. Někteří DNS poskytovatelé nabízí přístup přes rozhraní API pro jejich systémy. Bohužel firemní DNS poskytovatel nemá žádné API rozhraní a má zastaralou serverovou konfiguraci, takže není možné použít CURL nebo jiné technologie jako python nebo node. Jako první technologii, kterou student zkusil a neosvědčila se byla WebDriver.io, na které byly zdárně provedeny kroky, které vedly k automatickému nasazení validačního řetězce na server. Problém nastal při nasazování do Gitlab CI s headless prohlížečem *phantomjs*. Při spouštění pipeliny s vytvořeným řešení v Gitlab CI se nepodařilo spustit webdriverIO kvůli absenci selenia. I přes správně nainstalované balíčky pro phantomjs i to, že balíček *selenium-standalone* byl přidaný do souboru *package.json* v projektu. Kvůli tomuto problému zkoušel student změnit základní obraz pro kontejner z Debianu verze 9 na Node, ale při tomto postupu nebyl student schopen nainstalovat certbot. Tento problém vedl ke změně technologie potřebné k vypracování úkolu. Jako druhá technologie, která přišla na řadu byla Selenium-Webdriver. Selenium-Webdriver je nejznámější z open source nástrojů pro automatizaci ve webovém prohlížeči. Tento nástroj podporuje velké množství programovacích jako třeba Java, C#, Ruby, Python, a JavaScript. Pro tento nástroj je napsána podrobná dokumentace a má rozsáhlé rozhraní API. Selenium-Webdriver dokáže ovládat jakýkoliv nainstalovaný prohlížeč. Akorát pro

každý prohlížeč kromě FireFoxu musí být nainstalovaný speciální driver, který umožnuje propojení prohlížeče s WebDriver API rozhraním. [6]

Díky této změně byl student schopen spustit automatizaci v headless prohlížeči v pipelině Gitlab CI a zdárně nastavil DNS záznamy vytvořené v DNS výzvě na systém firemního DNS poskytovatele.

```
#!/bin/bash
CURRENT_PATH=$(pwd)
number_of_domains=2    # Change the value when you add/remove one or more
    domains
number=$(($number_of_domains*2))

echo $CERTBOT_DOMAIN >> $CURRENT_PATH/domain.txt
echo $CERTBOT_VALIDATION >> $CURRENT_PATH/domain.txt

count=$(wc -l < $CURRENT_PATH/domain.txt)

if [ $count -eq $number ];
then
    yarn start $DNS_login $DNS_password $count
    seconds=$(((3600 - $(date +%s) % 3600)+600))
    sleep $seconds
fi
```

Výpis 2: Authenticator.sh skript

Za zmínku stojí fakt, že auto-hook skript se spustí několikrát, pokud je nastaveno vytváření certifikátů pro více než jednu doménu. Z tohohle důvodu je definována proměnná number_of_domains, která udává maximální počet domén. Toto číslo se dále používá k rozpoznání stavu, kdy skript pracuje s poslední doménou. Tehdy se může spustit javascriptový kód, který s použitím Selenium-Webdriver nastaví řetězce výzev na DNS server poskytovatele a následně počká do další celé hodiny a deseti minut na ověření výzvy a poté se vrátí do CertBot skriptu.

Každá iterace authenticator.sh uloží CERTBOT_DOMAIN a CERTBOT_VALIDATION na nový řádek do vytvořeného textového souboru. Pokud se počet slov v textovém souboru rovná dvojnásobku počtu domén, pro které se vytváří certifikát, znamená to, že jsou vygenerovány všechny záznamy, které jsou potřebné pro nasazení na server DNS poskytovatele a spustí se javascriptový kód se třemi parametry, který se o tenhle úkol stará. Parametry, které se předávají, jsou: DNS_login, DNS_password (uložené proměnné na Gitlab CI) a počet domén, na které se vytváří SSL certifikát.

Na nahrání textových záznamů DNS výzvy byl vybrán jazyk Javascript a jeho rozšíření Selenium Webdriver z důvodu absence rozhraní API na straně DNS poskytovatele, a tak bylo nutné provést nasazení formou automatizace v prohlížeči.

Jako první v Javascript kódu je třeba načíst soubor s validačními řetězci a jejich jména, pod kterými mají být uloženy na straně DNS serveru. Zadruhé bylo třeba najít všechny bílé znaky pomocí regulárních výrazů a nahradit je prázdnými řetězci. Za třetí bylo potřeba zkontrolovat název validačního řetězce a přidat na začátek příponu *_acme-challenge*. Jako poslední úprava bylo rozdělení doménového jména z důvodu zadávání domény 1. a 2. řádu do jiného sloupce v tabulce TXT záznamů na straně DNS poskytovatele.3

```
let DNS_array = fs.readFileSync('domain.txt').toString().split("\n");
for(i in DNS_challenges_array) {
    DNS_array[i].replace(/\s+/g, '');
    if (i \% 2 == 0){
        if(DNS_array[i] == 'profiq.com')
        {
            DNS_array[i] = '_acme-challenge';
        }
        else{
            DNS_array[i] = DNS_challenges_array[i].slice(0, -11);
            DNS_array[i] = '_acme-challenge.' + '' + DNS_array[i];
        }
    }
}
```

Výpis 3: Úprava TXT záznamů

Po úpravě řetězců mohl student pomocí headless prohlížeče začít automatické přihlášení a zadání všech záznamů na stranu DNS poskytovatele. Automatizace probíhala pomocí hledání elementů podle jejich CSS, a to z důvodu absence ID elementů a duplicitě elementů pro názvy. Kód tedy vypadá následovně:4

```
    driver.findElement(By.css('.logintable:nth-of-type(2) [name=login]'))
        .then((el) => {
            el.sendKeys(login)
        });
    driver.findElement(By.css('.logintable:nth-of-type(2) [name="password"]'))
        .then((el) => {
            el.sendKeys(password)
        });
```

Výpis 4: Vyhledávání elementů podle CSS

4.1.4 Nasazení do Gitlab CI/CD

Jako poslední studentův úkol bylo nasazení vytvořeného řešení na Gitlab CI/CD pomocí YAML souboru.

Continuous Integration, nebo taky CI, funguje na integraci kódu z vašeho týmu ve sdíleném repozitáři. Vývojáři sdílí své nové kódy v Merge/Pull requestech, které spustí sestrojení, testy a CI zkontroluje nový kód před přidáním do repozitáře.

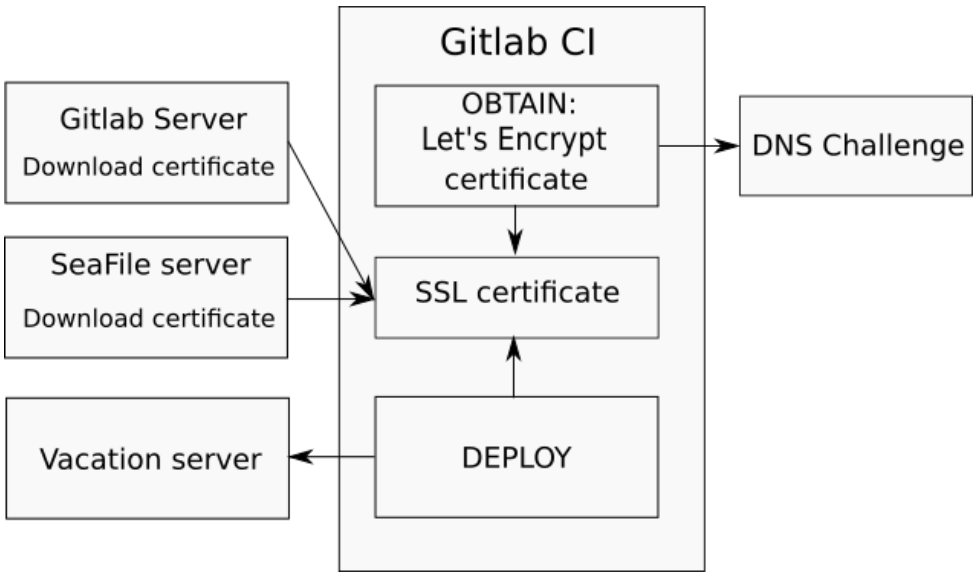
Continuous Delivery, nebo taky CD, dodává ověřený kód z CI do aplikace.

Společně CI a CD urychlují postup, jak může tým přinést výsledky. CI pomáhá zachytit a omezit chyby v brzkém vývojovém cyklu a CD přesune ověřené výsledky do aplikace.

Gitlab CI/CD pipeliny jsou nakonfigurovány použitím YAML souboru nazývaným .gitlab-ci.yml v každém projektu a běží jako samostatné kontejnery. Pro toto řešení vychází obraz z debianu verze 9 a pro přípravu potřebných balíčků se používají klasické linuxové příkazy. [7]

Jako první bylo potřeba si upřesnit, jakým způsobem se budou vytvořené certifikáty nasazovat na servery.

Proces se rozdělil na dvě fáze. Fáze pro vytvoření samotného certifikátu pomocí postupů popsaných v předchozích částech této práce a jeho zaheslování a uložení jako Gitlab artefakt a fáze pro nasazení na server pomocí SSH přístupu pro evidenci dovolených ve firmě. Zbylé dva servery nejsou zahrnuty ve fázi pro nasazování na server z důvodu bezpečnosti, ale mají nastavený Bash skript spouštěný v CRONu, který stáhne přes API poslední artefakt z Gitlab CI a provede nasazení a restartování serverů.



Obrázek 2: Rozdělení procesu na Gitlab CI/CD

Pro lepší představu finálního řešení je přiložený obrázek 2, na kterém jde vidět kompletní řešení ve zjednodušeném schématu. Celý proces běží v Gitlab CI, až na dva servery, které se na Gitlab CI napojují skrze API, aby si stáhly zkomprimovaný a zašifrovaný soubor s certifikátem,

který je uložený jako artefakt po vykonání první fáze pro vygenerování certifikátu (OBTAIN). Na poslední ze serverů se certifikát nasadí pomocí druhé fáze na Gitlab CI (DEPLOY), ve které se pomocí SSH připojíme na server a provedeme nasazení a restartování serveru.

4.2 Automatické testování backend API

Tato kapitola je zaměřená na popis řešení druhého studentova úkolu na vytvoření automatického testování backend API. V jednotlivých částech této kapitoly je popsáno, jak student postupoval, na jaké problémy v tomto procesu narazil při vypracování tohoto úkolu a jak je řešil.

4.2.1 Seznámení s projektem

Jako první úkol pro studenta bylo jeho seznámení s projektem, pro který měl vytvořit automatické testování. Jedná se o backend firemního protokolu LDAP, který řeší, jak jsou jednotlivé položky na serveru ukládány formou záznamů a uspořádány do stromové struktury. Protokol LDAP stojí nad TCP/IP a umožnuje klientům provádět různé operace v adresářovém serveru, včetně ukládání a získávání dat, hledaní dat podle zadaných kritérií, ověřování klientů a podobně.

Tento backend se vytvoří tak, že se vytvoří Docker kontejner a následně se spustí spolu s Nginx kontejnerem, který ho přesměruje z portu 8080 na port 80 na api.ldap.profiq.com. Což je místo, kde student bude provádět testování.

Další krok bylo zvážit, jaké testy by bylo vhodné vytvořit a nasadit do automatického testování. Mezi základní testovací scénáře bylo přidáno testování různých variant přihlašování, změna hesla, přidání nového uživatele a smazání uživatele. Mezi další funkcionalitu, kterou bylo nutné otestovat, byl reset hesla uživatele, kde server pošle nové heslo na email uživatele. Na konec bylo potřeba otestovat pravomoce pomocí atributů, kterých uživatele můžou nabývat.

4.2.2 Provedení manuálního průzkumného testování

Před začátkem samotného automatizování bylo nutné provedení manuálního testování, aby se zjistilo, zda bude možno všechny scénáře zhotovit. Veškeré manuální testování bylo prováděno pomocí softwaru Postman, z důvodu nutnosti otestování stavových kódů při odpovědi serveru na klientský požadavek. Při manuálním testování student sepisoval testy do jednodušších částí a dopisoval k nim očekávaný výstup, např. při zadání špatného hesla při přihlášení bude vrácen stavový kód 400. Další krok bylo sepsání priorit k jednotlivým částem testování. Větší prioritu mají části, které jsou často používané a jsou klíčové ke správnému chodu aplikace.

Postman (software) je nástroj, který je užitečný, když je potřeba prozkoumat REST API vytvořené někým jiným nebo otestovat vámi vytvořené. Tento nástroj má elegantní uživatelské rozhraní, pomocí kterého je možné vytvářet HTTP požadavky bez psaní spousty kódů a hodí se výborně na testování rozhraní API.

4.2.3 Vytváření automatických testů

Jelikož se v tomto úkole jedná o funkční testování, bylo potřeba vybrat vhodný nástroj, který podporuje vytváření HTTP požadavků, protože se v tomto úkolu budou testovat stavové kódy pro odpovědi serveru na klientské požadavky. Pro tento úkol byl vybrán programovací jazyk Python s rozšířením unittest a requests na vytváření HTTP požadavků. Dále bylo doporučeno studentovi si vyzkoušet aplikaci Postman před začátkem automatizace testů.

unittest je testovací framework, který byl původně inspirovaný JUnit frameworkem a funguje na podobné bázi jako ostatní testovací frameworky v ostatních jazycích. Tento framework podporuje automatizaci testů, sdílení nastavení a vypínání kódu pro testy, agregaci testů do kolekcí a nezávislost testů.

requests je knihovna, která je určena pro vytváření HTTP požadavků. Přestože HTTP požadavky jde vytvářet i bez pomocí dodatečných knihoven, a to pouze se standardní knihovnou v Pythonu, bylo pro toto řešení vybráno pracovat s dodatečnou knihovnou requests. Důvod k tomuto kroku bylo převážně to, že knihovna requests má mnohem lidštější rozhraní a používá se podstatně jednodušeji.

Jako první bylo nutné vytvořit spustitelný Python soubor s unittest knihovnou.5

```
import unittest
import requests
class TestStringMethods(unittest.TestCase):

    def test_login_options(self):
        test = requests.options(url_login)
        self.assertEqual(test.status_code, 200)
   ...
if __name__ == '__main__':
    unittest.main()
```

Výpis 5: Základní spustitelný soubor pro testování

Další krok pro studenta bylo vytvoření konfiguračního souboru se všemi proměnnými, autentizačními hodnotami, URL adresami, JSON soubory pro různé testovací případy a hlavičkami. Tento konfigurační soubor byl vytvořen z důvodu zamezení duplicity záznamů a pro jednodušší přepisování záznamů v případě potřeby.

První sada testů, která kontroluje základní funkce projektu, byla napsána studentem za krátkou dobu, kde díky předem vytvořenému konfiguračnímu souboru bylo lehké vyzkoušení například přihlašování s různými kombinacemi správných a špatných přihlašovacích údajů. Všechny kombinace byly připraveny jako JSON v konfiguračním souboru.

První problém nastal, když student potřeboval otestovat resetování hesla. Kvůli tomuhle testu byl student nucen vytvořit funkci, která se přihlásí na testovací email a z posledního

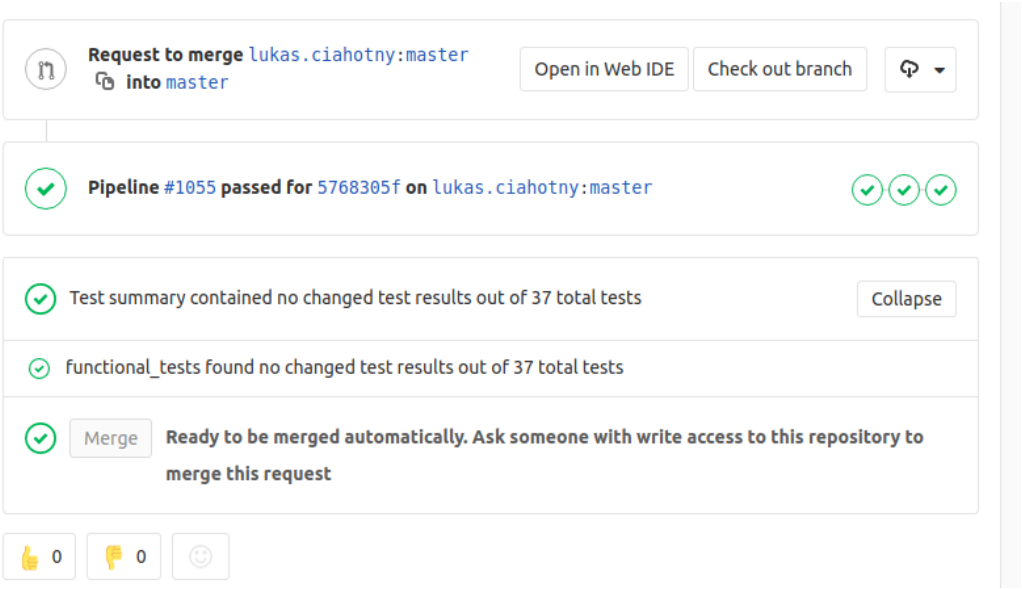
přijatého emailu vybere pomocí regulárních výrazů nově vygenerované heslo, které bylo uloženo jako návratová hodnota funkce.6

```
def read_email_from_gmail():
    try:
        mail = imaplib.IMAP4_SSL(SMTP_SERVER)
        mail.login(FROM_EMAIL,FROM_PWD)
        mail.select('inbox')

        type, data = mail.search(None, 'ALL')
        mail_ids = data[0]

        id_list = mail_ids.split()

        for i in reversed(id_list):
            typ, data = mail.fetch(i, '(RFC822)')
            for response_part in data:
                if isinstance(response_part, tuple):
                    msg = email.message_from_string(response_part[1].decode('utf
                        -8'))
                    string = msg._payload
                    email_message = re.search( r'Your new password is :<\/b> (.*)
                        <\/p>', string, re.M|re.I)
                    if email_message:
                        return email_message.group(1)
                    else:
                        return "None"
            break

    except Exception as e:
        print(e)
```

Výpis 6: Funkce pro čtení emailů

Další problém spočíval v kontrolování atributů uživatele. Bylo nutné vytvořit funkci, která bude mít na vstupu uživatele a atribut, který se má najít. Ve funkci bylo prvně potřebné nahradit jednoduché uvozovky za složené a následně dekódovat JSON soubor. Poslední dvě funkce, které student vytvořil, byly na kontrolu povolených atributů, které uživatel může mít na kontrolu pravomocí uživatele.

Poslední úkol, který měl student vypracovat v tomto zadání, bylo zavedení testu na gitlab CI a pomocí JUnit test reports [9] a xmlrunner nastavit spuštění vytvořených testů při každé změně



Obrázek 3: Výsledek v Gitlabu

kódu. Jako první bylo nutné změnit main v souboru s testy, jak je možné vidět na obrázku.7

```
if __name__ == '__main__':
    with open('results.xml', 'wb') as output:
        unittest.main(
            testRunner=xmlrunner.XMLTestRunner(output=output),
            failfast=False, buffer=False, catchbreak=False)
```

Výpis 7: Main pro xmlrunner

A nakonec přidat **junit: results.xml** do YAML souboru v Gitlab repozitáři projektu. [10] Díky tomuto nastavení se vytvořené testy spustí pro každý Pull Request (PR) ve verzovacím systému Git. A díky využití knihovny XMLTestRunner jsou při vyskytnutí chyby informováni jak tester, tak programátor, který aplikaci upravil. Jako vedlejší efekt nižší chybovosti jsou nižší náklady na vývoj [11], jelikož není nutné chyby opravovat. S aplikací bez chyb jsou spokojení i uživatelé. Finální výsledek práce při Merge requestu je možné vidět na obrázku.3

5 Teoretické a praktické znalosti a dovednosti získané v průběhu studia a uplatněné v průběhu praxe

Student během vypracování bakalářské praxe využil znalosti nabyté ze všech programovacích předmětů. Především informace o koncepci a stylu programování studentovi pomohly psát kód lehce rozšiřitelný a přehledný. Před absolvováním bakalářské praxe studentovi velmi pomohly předměty jako Skriptovací programovací jazyky a jejich aplikace, protože v tomto předmětu se programovalo v Pythonu, který student využil v průběhu své praxe na obou svých úkolech. Dále získal základní znalosti Linuxu v předmětu Architektury počítačů a paralelních systémů, kde se používal operační systém Linux. Linux byl totiž hlavní platformou, na které student během své praxe pracoval.

Student v průběhu odborné praxe pracoval s programovacím jazykem Javascript, a proto pro něj byl přínosný předmět Tvorba aplikací pro mobilní zařízení I., ve kterém se programovalo právě v programovacím jazyku Javascript a díky této znalosti se student rychleji zorientoval při zpracování svých úkolů.

Jako další přínos pro studenta byly hodiny angličtiny v průběhu čtyř semestrů (Ab/I-FEI - Ab/IV-FEI), protože veškerá dokumentace a tutoriály, které si student musel prostudovat, byly napsány právě v anglickém jazyce.

6 Znalosti či dovednosti scházející studentovi v průběhu odborné praxe

Schopnost nejvíce scházející studentovi byla zkušenost s verzovacími systémy. Při prvním setkání s technologií Git studentovi zabralo značnou chvíli, než se naučil, jak manipulovat s různými větvemi stejného kodu, jejich spojování a celkově nastavit Git na lokálním počítači. Verzovací systémy se dnes používají běžně v praxi, a to i při malých projektech, a tudíž by bylo přínosné pro studenty, kdyby se s verzovacími systémy vice seznámili při studiu.

Jako další znalost, která studentovi scházela v průběhu odborné praxe, byla neznalost technologie Docker, která se používá čím dál častěji v praxi pro izolaci procesů a jejich virtualizaci. Díky této neznalosti byl student nucen se s technologii Docker seznámit během praxe, a tudíž zpomalila studentův počáteční pracovní výkon.

7 Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

7.1 Výsledky studentovy práce

Hlavní cílem této práce bylo popsat studentovy činnosti na projektech, které vykonal během bakalářské praxe formou individuální odborné praxe.

Výsledkem studentovy práce bylo vytvoření automatického procesu běžícím na firemním Gitlab CI pro vytváření Let's Encrypt certifikátu na subdomény v intranetu a jejich následné nasazování na servery, celkem na 3 subdomény. Celkové řešení je napsané tak, aby bylo lehce rozšiřitelné na větší počet subdomén v případě potřeby.

Taktéž byly vytvořeny testy na automatické testování backend API pro aplikaci na evidování dovolených ve firmě. Testy jsou napsány v pythonu v knihovně unittest a jsou spuštěny pokaždé, když nastane požadavek na merge request do hlavního repozitáře na Gitlabu. Celkem bylo vytvořeno 37 testů, kde některé využívají dodatečných funkcí, které student musel napsat, aby bylo možné řádně zkontrolovat výsledky.

Studentem vytvořené automatizace se osvědčily a jsou používány pravidelně v případě vytváření a nasazování SSL certifikátu. Tento proces se opakuje jednou týdně a zabezpečuje komunikaci na firemním intranetu. U druhého úkolu na automatické testování backend API jsou spuštěny testy při každém merge requestu do hlavního repozitáře a pomáhají předejít chybám v základní funkcionalitě aplikace před mergnutím do hlavního repozitáře.

7.2 Zhodnocení odborné praxe

Během praxe si student rozšířil své vědomosti v programování v jazycích Javascript, Python a skriptování v Linux Bash. Taktéž se naučil pracovat s operačním systémem Linux a pracovat s technologii Docker. Dále si student vyzkoušel pracovat s verzovacím systémem Git, který se běžně používá v praxi. Dále si student zlepšil komunikaci v angličtině, kterou používal v písemné i ústní komunikaci.

Po odpracování dnů v rámci bakalářské praxe zůstal student ve firmě a zařadil se mezi běžné zaměstnance a byl přiřazen do komerčního projektu na poloviční úvazek.

Literatura

[1] About us » profiq . *profiq* [online]. Copyright © 2019 profiq inc. All rights Reserved. [cit. 11.04.2019]. Dostupné z:
`https://www.profiq.com/about-us/`

[2] Jiří PETERKA *Báječný svět elektronického podpisu*. CZ.NIC, 2011, ISBN 9788090424838.

[3] Getting Started - Let's Encrypt - Free SSL/TLS Certificates. *Let's Encrypt - Free SSL/TLS Certificates* [online]. [cit. 11.04.2019] Dostupné z:
`https://letsencrypt.org/getting-started/`

[4] User Guide — Certbot 0.34.0.dev0 documentation.*Certbot* [online]. Copyright © Copyright 2014 [cit. 11.04.2019]. Dostupné z:
`https://certbot.eff.org/docs/using.html#manual`

[5] Challenge Types - Let's Encrypt - Free SSL/TLS Certificates. *Let's Encrypt - Free SSL/TLS Certificates* [online]. [cit. 11.04.2019]. Dostupné z:
`https://letsencrypt.org/docs/challenge-types/`

[6] Selenium Documentation — Selenium Documentation. *Selenium - Web Browser Automation* [online]. Copyright © Copyright 2008 [cit. 11.04.2019]. Dostupné z:
`https://www.seleniumhq.org/docs/`

[7] GitLab CI/CD Pipeline Configuration Reference | GitLab. *GitLab Documentation* [online]. [cit. 11.04.2019]. Dostupné z:
`https://docs.gitlab.com/ee/ci/yaml/`

[8] Building Docker images with GitLab CI/CD | GitLab. *GitLab Documentation* [online]. [cit. 11.04.2019]. Dostupné z:
`https://docs.gitlab.com/ee/ci/docker/using_docker_build.html`

[9] JUnit test reports | GitLab. *GitLab Documentation* [online]. [cit. 11.04.2019]. Dostupné z:
`https://docs.gitlab.com/ee/ci/junit_test_reports.html`

[10] GitHub - xmlrunner/unittest-xml-reporting: unittest-based test runner with Ant/JUnit like XML reporting.. *The world's leading software development platform · GitHub* [online]. Copyright © 2019 [cit. 11.04.2019]. Dostupné z:
`https://github.com/xmlrunner/unittest-xml-reporting`

[11] Miroslav BUREŠ, Miroslav, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ *Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu.*. Praha: Grada, 2016, ISBN 9788024755946.